EQUIP: an Extensible Platform for Distributed Collaboration

Chris Greenhalgh,
Mixed Reality Laboratory, School of Computer Science and IT,
University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
Email: cmg@cs.nott.ac.uk

Abstract

EQUIP is a new software platform for data sharing and collaboration being developed within the EQUATOR Interdisciplinary Research Collaboration (IRC) in the UK. EQUIP has a highly modular software organization, incorporating dynamic code loading and multi-language support (currently C++ and Java). It is also designed to run on a wide range of devices, from handhelds to graphical supercomputers, and has support for both wired and wireless networking. Core to many EQUIP applications is its data sharing service, which combines work on Collaborative Virtual Environments with ideas from tuple-spaces, to create an open, extensible and interactive data sharing platform for general networked and collaborative applications. Coupled with EQUIP's code-loading framework this support distribution of computation as well as communication.

1. Introduction.

EQUIP [www.equator.ac.uk/technology/equip] is a new software platform for data sharing and collaboration being developed within the EQUATOR Interdisciplinary Research Collaboration (IRC) in the UK [www.equator.ac.uk].

Much of the pre-history of EQUIP is in the area of Collaborative Virtual Environments (CVEs), which are networked multi-user virtual environment designed to support communication and collaboration through a combination of media, typically including text, audio, video and 3D graphics. In particular it builds on the experiences of building and using the MASSIVE-1, 2 and 3 CVE systems [www.crg.cs.nott.ac.uk/~cmg]. All of these systems have been used in quite extensive field trials with significant numbers of (expert and non-expert) users.

1.1. Goals.

In designing and developing EQUIP we have sought to build on this experience, but also to address a number of specific goals:

- highly modular software organisation, to support maintainability and extensibility;
- run-time code loading, for extensibility and code mobility and distribution;
- cross language to give access to development environments with higher performance (C/C++) and higher programmer productivity (Java).

In addition, within the EQUATOR IRC we are increasingly interested in collaboration and distributed interaction through mobile technologies, such as handheld and wearable devices. Consequently EQUIP must also support:

- deployment on a range of hardware platforms, from handhelds to supercomputers (e.g. it has been used on platforms ranging from an iPAQ to an ONYX2 with multi-pipe Infinite Reality graphics for rendering in a ReaCTor);
- support for both wired and wireless networking, and the resultant variability in bandwidth and connectivity.

2. Overview.

We have been developing EQUIP since January 2001, and the current version (which is freely available from a public CVS repository – see [http://www.crg.cs.nott.ac.uk/~cmg/Equator/]) comprises:

- A cross-platform build system, which is Bamboo's [www.watsen.net/Bamboo].
- A portable run-time layer, currently Netscape Portable Runtime (NSPR) for C++ and any JDK (version 1.1.3 or above) for Java.

- A code loading and modularisation layer, which is Bamboo for C++ and (currently) standard code loaders and packages for Java.
- Core runtime libraries supporting standard facilities such as memory management (for C++), serialisation (dual language) and simple TCP/UDP networking.
- A language independent type-definition language and code generator based on CORBA IDL valuetypes. This provides code generation for cross-language operations including serialisation, equality testing and pattern matching.
- A shared data service, which incorporates ideas from tuplespaces with our previous work on efficient and timely data sharing for CVEs.
- A large set of modules (currently about 60) which include standard cross-language data types (e.g. for maths) and various clients and servers (e.g. an extensible 3D renderer which uses VRJuggler [www.vrjuggler.org] and OpenGL).

This is illustrated in figures 1 and 2.

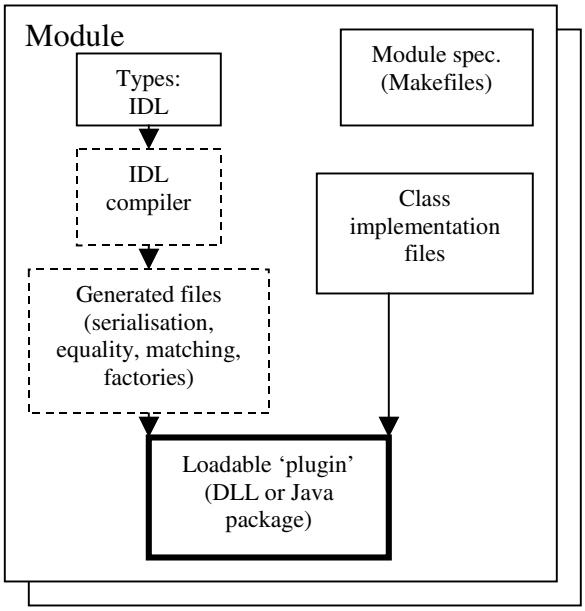

Figure 1. Development time structure of an EQUIP module.

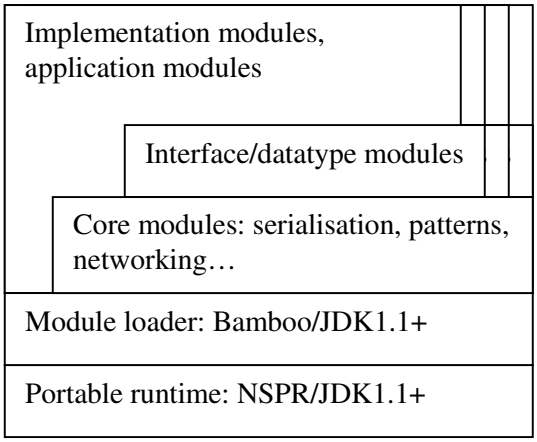

Figure 2. Runtime structure of EQUIP.

3. EQUIP in Use.

Most applications of EQUIP to date revolve around its shared data service. This supports easy authoring of loosely coupled distributed systems for a range of applications (based on pattern matching and application-specific object definition). There is also increasing support for wireless access.

Its performance can directly support interactive applications such as simple shared scene graphs for CVEs. To support higher performance applications (e.g. streaming audio and video, complex visualizations) the data service is used to share meta-data (e.g. session descriptions, database references) and/or application-specific dynamically loaded objects, which can then handle local/distributed computation, and communicate using other more streamlined communication mechanisms.

3.1. Collaborative Visualisation Example.

As an example of the potential use of EQUIP, we describe a simple collaborative visualisation session which was staged using EQUIP between a user in the Reality Centre at Nottingham and a user in the 4-wall ReaCTor at UCL in London in February 2002. The arrangement of machines and processes was as shown in figure 3.

A single master dataspace was run at Nottingham, and used to share objects including representations of user avatars and visualization elements. Each site ran a renderer as a client of this data space. The renderer – which uses VRJuggler to handle window configuration and input – requests all 3D drawable objects from the dataspace. It forms these into an implied scene graph which it traverses, asking each object to render itself in turn. For some objects, such as the user's avatars, this is a

simple geometry blob, for example read from a URL. For other objects, such as visualization elements, this can involve additional computation and dynamic rendering. In this example a simple 3D scatter plot of database tables was renderered by the class equip.draw3d.pit.Object3DPit. Note that the code implementing this class – including its rendering – was dynamically loaded by each renderer process when the object was first replicated from the dataspace. Dynamic updates to the shared state allow the visualization to be manipulated.

The EQUIP data service can also be used to share other kinds of information. In this example, the user at Nottingham had no tracking device with which to navigate viewpoint of the Reality Centre; instead they used an EQUIP client to simulate a joystick on another machine, and published this information in the same EQUIP dataspace. A custom VRJuggler input device was then used to take this information published in EQUIP and present it within VRJuggler as a normal analogue input device to control navigation. The joystick simulator can be transparently substituted with a client publishing real joystick information, or with a client running on a wireless handheld device (e.g. iPAQ with WaveLAN card); we have demonstrated both of these alternatives in EQUIP.

This trial used streaming IP audio based on the MASSIVE audio service, which can be controlled via session information shared in an EQUIP dataspace.

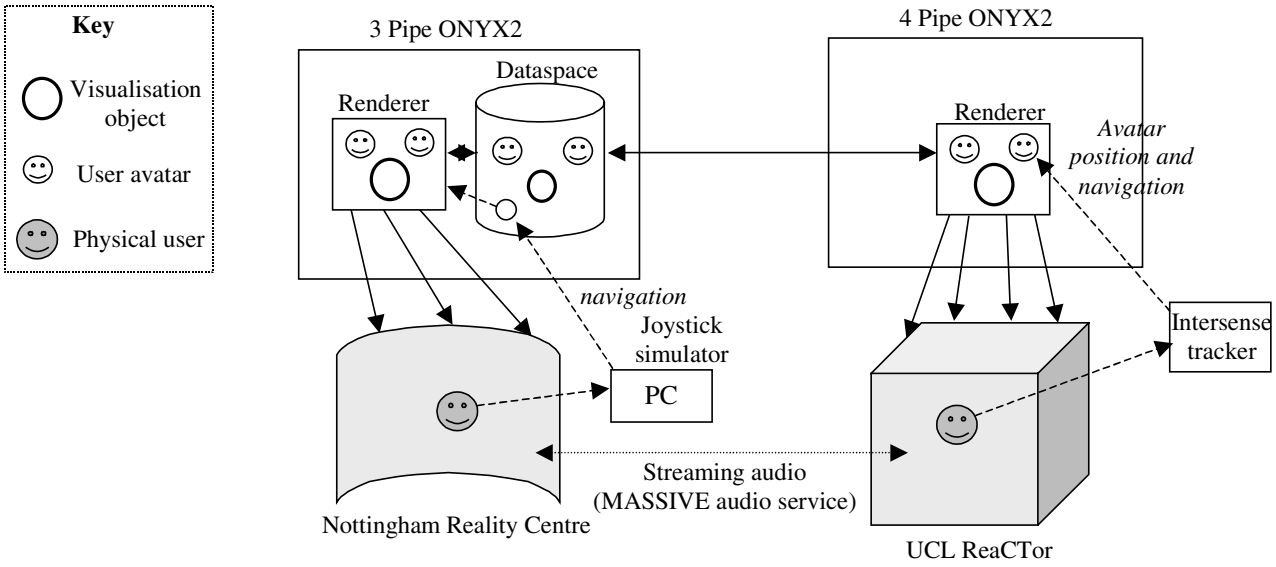

Figure 3. Collaborative Visualisation Example.

4. Future Directions.

EQUIP is under continuous and ongoing development. At present the core development team is small, although the user community is somewhat larger (it is used in several sub-projects within the EQUATOR IRC, and also in a number of other projects that involve members of the IRC).

Areas for shorter-term development include:

- Better support for wireless use and intermittent connectivity.
- Support for multi-process application configuration, deployment and monitoring;
- More general framework for sensing and (physical) environment modelling, e.g. for context-driven interaction.
- Support for one or more open scene graphs, and for more sophisticated rendering and interaction techniques, e.g. multipass rendering.

5. Acknowledgements.

We would like to thank the UK's Engineering and Physical Sciences Research Council, for their support of the EQUATOR IRC. We would also like to thank the members of IRC – and others – who have contributed to EQUIP, both through development and use.